ARPA ORDER NO. 189-1

RM-6338-ARPA
AUGUST 1970

A DYNAMIC PROGRAMMING APPROACH TO NETWORK PROBLEMS: A MODEL FOR ON-LINE COMPUTER SYSTEMS

L. J. Pipes

prepared for
ADVANCED RESEARCH PROJECTS AGENCY

Rand
SANTA MONICA, CA. 90406

ARPA ORDER NO. 189-1

RM-6338-ARPA
AUGUST 1970

A DYNAMIC PROGRAMMING APPROACH TO NETWORK PROBLEMS: A MODEL FOR ON-LINE COMPUTER SYSTEMS

L. J. Pipes

This research is supported by the Advanced Research Projects Agency under Contract No. DAHC15 67 C 0142. Views or conclusions contained in this study should not be interpreted as representing the official opinion or policy of Rand or of ARPA.

Rand
SANTA MONICA, CA. 90406

PREFACE

The broad range of optimization problems that can be classified as multistage decision processes are amenable to solution by the techniques of dynamic programming. This Memorandum provides a specific application of the dynamic-programming approach to solving such problems, which arise in many areas of scientific investigation.

The present study is an outgrowth of an investigation of border infiltration undertaken by The Rand Corporation for the Advanced Research Projects Agency. In that investigation, the technique presented here was applied to the more limited problem of predicting infiltration routes. The model, however, has now been generalized so that it can be applied to a variety of problems.

This model is programmed for use on JOSS, Rand's time-shared, on-line computer system, but it is designed to be readily adaptable to other on-line systems. At present, JOSS is being used by approximately 20 agencies within the Department of Defense and by numerous other organizations, as well as by Rand. And since all of these agencies deal with practical problems that are structured with many decision variables and constraints, this model should prove widely useful.

SUMMARY

This Memorandum contains the derivation of a dynamic-programming model for finding optimal solutions to problems involving certain multi-stage decision processes. The model has been implemented on JOSS, and instructions on its use are also included.

The dynamic-programming approach analyzes an optimization problem with various constraints and variables by decomposing the problem into a sequence of stages at which lower-order optimization takes place. The model presented here encompasses a value-iterative method which is less restrictive and which offers desirable advantages over other currently used techniques. One such advantage is a reduction in the actual amount of computer storage required to reach an optimal solution.

The model is flexible and allows for testing the sensitivity of a decision process to changes in the terminal point and thus in the associated costs. Additional information may be gained from examining the buildup of an optimal solution, which can also be printed out if conditions permit.

ACKNOWLEDGMENTS

It is a pleasure to acknowledge the helpful comments of C. F. Black, J. A. Lockett, and K. V. Saunders.

CONTENTS

I. INTRODUCTION

The term *dynamic programming* is frequently applied to the mathematical analysis of problems in which conditions that must be satisfied by an optimal time-staged decision process are to be exploited to determine the best course of action. Dynamic elements (i.e., simultaneous time considerations) are foremost in many types of problems involving multistage decisionmaking over definite--or even indefinite--time horizons.*

Some of the subtle points of dynamic programming are frequently obscured in discussions of the subject. In Section II, we shall attempt to clarify these points, as they relate to a class of decision problems that can be characterized by a particular network system. This, then, facilitates the construction of a model suitable for finding optimal solutions to such problems. The principle of optimality used in the present model allows one to find the minimum-cost (shortest) path as a function of the maximum number of arcs allowed. (The unconstrained shortest-path problem has been treated extensively, e.g., see Refs. 1 and 2.)

Section III describes the JOSS** implementation of the model and includes instructions for its use. An illustrative problem is given in Appendix A, and the program listing is given in Appendix B.

---

*A definite time horizon is one in which the number of periods or stages remaining in a decision process is stipulated.

**JOSS is an on-line, time-shared computer system developed at Rand. JOSS is the trademark and service mark of The Rand Corporation for its computer program and services using that program.

II. THE MODEL

The class of problems with which we are concerned is defined as follows: Let a decision process be characterized by a network system consisting of p nodes and a collection of directed arcs. (A node is merely a junction point with incoming and/or outgoing arcs.) Let (i,j) denote the arc from node i to node j, and the amount C_{ij} be the associated expense of traversing (i,j). A terminal node, r, must be specified. There is, however, no requirement that a particular starting point be specified.

Given a decisionmaking process that can be formulated by such a network system, we define an optimal solution to be a path (from nodal points in the system to the terminal) that satisfies the following principle of optimality:

> *Regardless of the previous rationale used to arrive at a particular state, the remaining decisions as to what path to take en route to the terminal must themselves constitute an optimal solution.*

Here we let Y_i be the present value of an optimal path from node i to the terminal node r, where *optimal* implies a path having minimal total cost. Now, if an optimal path from node i to node r starts by first going to node j, then

$$Y_i = Y_j + C_{ij} \tag{1}$$

and

$$Y_i \leq Y_k + C_{ik} \tag{2}$$

for all $k \neq j$ in the system.

Equation (1) states that the cost of an optimal path from node i to the terminal, r, which starts by first going to node j is the sum of the cost of an optimal path from node j to r and the cost of going from node i to node j. Equation (2) insures that among the possible

choices of paths to take from node i, none is better than the one starting with node j.

Since Eqs. (1) and (2) must hold for every node i different from the terminal node r, then Y_i must satisfy a set of functional equations,

$$Y_i = \underset{\substack{(i,j) \\ \text{in the system}}}{\text{minimum}} [Y_j + C_{ij}] \tag{3}$$

for all $i \neq r$, and

$$Y_r = 0. \tag{4}$$

A method of successive approximation may be used to solve for the $p - 1$ unknowns, Y_i $(i \neq r)$. Specifically, a value-iterative algorithm is applied. The value of the Y_j's is initially set at zero, and the Y_i's are then calculated as in Eq. (3) for every node $i \neq r$ in the system. That is, the first iteration is computed as follows:

$$Y(1,i) = \underset{\substack{(i,j) \\ \text{in the system}}}{\text{minimum}} [0 + C_{ij}], \tag{5}$$

for all $i \neq r$ with $Y(1,r) = 0$. The first index on Y is the iteration number.

On the n^{th} iteration, the Y_i's computed during iteration $n - 1$ become estimates of the Y_j's, and new Y_i's are computed as follows:

$$Y(n,i) = \underset{\substack{(i,j) \\ \text{in the system}}}{\text{minimum}} [Y(n-1,j) + C_{ij}], \tag{6}$$

again for all $i \neq r$ and $Y(n,r) = 0$.

The quantity Y(n,i) is interpreted as the minimum cost of a path starting from node i that contains exactly n arcs, unless the terminal is on the path, in which case the path terminates at node r. If the cost, C_{ij}, around every loop in the system is positive, then an n^* exists

such that for all $n \geq n^*$, $Y(n,i) = Y(n^*,i)$. We terminate the iterative process when this condition is met. That is, after each iteration > 1, we test $Y(n,i) = Y(n-1,i)$, and when this condition is met for all i in the system, we have obtained the optimal solution. We also obtain another vector, g(n,i), which has its value set equal to the j that minimizes Eq. (5) or (6), depending on the iteration number.

That the process will terminate in a finite number of steps is shown by the following argument: As n increases, eventually every path of n arcs reaches the terminal. And when this occurs, Y(n,i) will stabilize to the correct minimal-cost value for the path, since the only new value added to Y(n,i) will be $Y(n,r) \equiv 0$.

Equations (5) and (6) form the basis of the model, which has been implemented on JOSS. The value-iterative algorithm is less restrictive and offers certain advantages over various other techniques which might be used. For example, the model does not require that the number of stages or nodes between the starting point and the terminal be stipulated. There is a *recursive algorithm* which is sometimes used and which, under our formulation, would compute an optimal solution from a particular node i with n more nodes between i and the final decision point. However, it could be that if the terminal is node 6 and the state of the system is at, say, node 5, the optimal path to the terminal might go through nodes 4, 1, 8, and 7, rather than going directly to 6 from 5. Similarly, a *policy-iterative* algorithm would require that a realizable path from node i to the terminal be stipulated before an optimal solution could be found.

Our model does not restrict the direction in which the arcs connecting nodes may go (i.e., paths may be cyclic), and it allows varying numbers of arcs to lead out of and/or into each nodal point. Section III illustrates these concepts and gives instructions on the use of the model.

III. THE JOSS PROGRAM

The JOSS program is stored in file 238(SYOP8) under Item 1 (DYNAM). After recall, the user types "Do part 1." to enter all necessary parameters of the system. Input includes the total number of nodes, the lowest node number, and the terminal node number of the system. Both consecutive and nonconsecutive numbered nodes are permitted. In the latter case, the model requests a number for each node in the system. The number of arcs leading out of each node is demanded as well as the node into which the j^{th} arc out of node i goes.

Costs for traversing (i,j) may be entered in part 1 or recalled from another stored item before operation of the program is initiated. If they are recalled from another item, they must be arrayed in the vector c(i,j). Alternatively, one might define c(i,j) to be a function by using a "Let" statement. The program will accept either procedure.

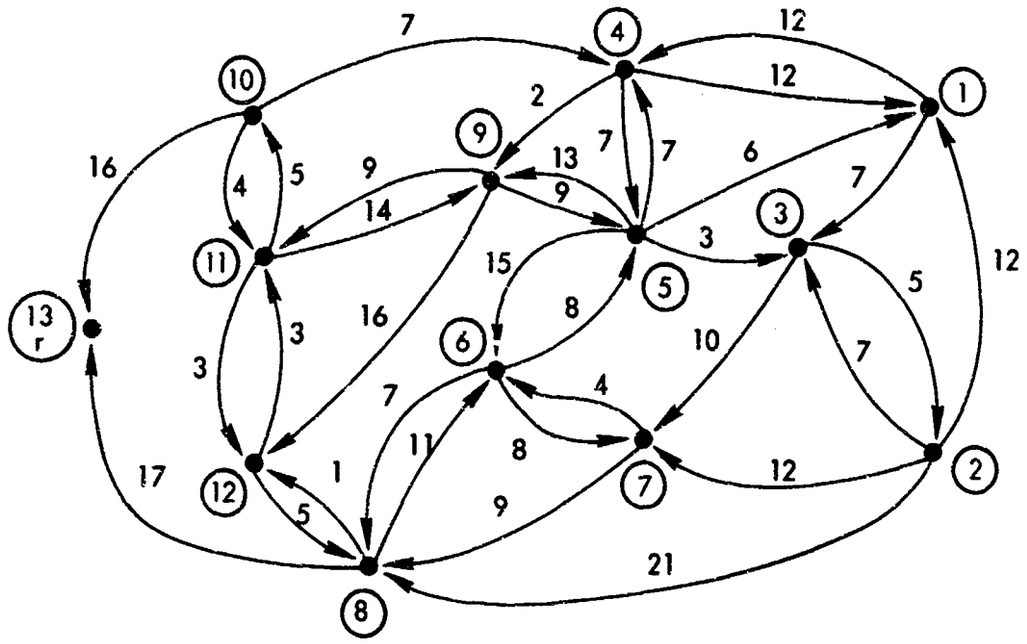
The program goes on to part 2, which controls the iteration algorithm. Since the model needs only to compare Y(n,i) and Y(n - 1,i) to determine whether or not an optimal solution has been reached, steps are included that delete all other stored present cost values and parts of the program when size requirements approach computer memory capacity. Given this feature, the model should be able to handle many problems which would otherwise require large amounts of storage space. However, it might first be necessary to delete y and g (present cost values are stored in array y, minimizing arcs are stored in array g) and then recall the program before reusing it, if deletion occurs. Both y and g are sparse arrays.†

The optimal solution is output as the best arc to take from each node en route to the terminal. The total minimum cost from each node to the terminal is also given, and if storage space permits, the build-up of the solution can also be output (i.e., the results from each iteration $< n^*$).

---

†JOSS initially considers all elements of these arrays to be zero, which facilitates our first estimate of the Y_j's.

As an illustrative example, consider the following network system:



Node numbers are circled; node 13 is the terminal. The costs for traversing the various arcs are shown alongside each arc. Any node might be the current state, node 13 being the state at the end of the final decision. To illustrate, a path along the arcs (1,3), (3,7), (7,8), and (8,13) represents the current decision to go from node 1 to node 3, the subsequent decision to go from node 3 to node 7, etc.

In a general sense, the network represents the possible connections between decisions to be made and the associated cost of going from one decision to another. For example, the different arcs into node 9 might represent alternative decisions in buying or selling.

The program requires 11 iterations to solve the network. The solution is given at the top of the following page.*

*See Appendix A for the input, optimal solution, and solutions buildup to this network. In the example of Appendix A, the terminal point was changed to node 5 and the optimal solution recomputed.

Best Path		Minimum Cost to
From	To	Terminal
1	3	43
2	8	38
3	7	36
4	9	32
5	6	39
6	8	24
7	8	26
8	13	17
9	11	30
10	13	16
11	10	21
12	8	22

Notice that this solution satisfies the principle of optimality defined earlier.

To investigate how the optimal set of nodes changes with changes in the terminal point, we simply set T equal to the desired node and type "Do part 2.". If only costs are to be changed, type "Do part 25.".

Each node which has at least one exit arc is included in the solution. In case of dead ends (i.e., nodes with no exit arcs), an attempt has been made to program around them by assigning a very high cost.

By the inclusion of random elements, the model could be extended to take into account the impact of uncertainty on an optimal decision process (this extension has not yet been made). In such cases, uncertainties are viewed as imperfect predictions to be characterized by probability distributions for the different possible events. Stochastic versions of many dynamic-programming models are often not much more difficult to solve than their deterministic counterparts. In the present case, such a model would be solving for a path having the least expected cost (i.e., the "most probable" path).

Appendix A

ILLUSTRATIVE PROBLEM: INPUT AND OUTPUT

```
Use file 238 (SYOP8).
Roger.
Recall item 1 (DYNAM).
Done.
Do part 1.
Number of nodes in the system = 13
Lowest node number in the system = 1
Terminal node number = 13
Are node numbers consecutive? (Answer:Yes=1;No=2) = 1
Into array A(i), enter number of arcs leading out of node i.
        A(1) = 2
        A(2) = 4
        A(3) = 2
        A(4) = 3
        A(5) = 5
        A(6) = 3
        A(7) = 2
        A(8) = 3
        A(9) = 3
       A(10) = 3
       A(11) = 3
       A(12) = 2
       A(13) = 0
Into array O(i,j), enter node number into which the j-th arc
out of node i goes.
      O(1,1) = 3
      O(1,2) = 4
      O(2,1) = 1
      O(2,2) = 3
      O(2,3) = 7
      O(2,4) = 8
      O(3,1) = 2
      O(3,2) = 7
      O(4,1) = 1
      O(4,2) = 5
      O(4,3) = 9
      O(5,1) = 3
      O(5,2) = 1
      O(5,3) = 4
      O(5,4) = 9
      O(5,5) = 6
      O(6,1) = 5
      O(6,2) = 7
      O(6,3) = 8
      O(7,1) = 6
      O(7,2) = 8
      O(8,1) = 6
      O(8,2) = 12
```

```
      O(8,3) = 13
      O(9,1) = 5
      O(9,2) = 11
      O(9,3) = 12
     O(10,1) = 4
     O(10,2) = 11
     O(10,3) = 13
     O(11,1) = 9
     O(11,2) = 10
     O(11,3) = 12
     O(12,1) = 11
     O(12,2) = 8
Are costs to be entered here? (Yes=1;No=2) = 1
Into array c(i,j), enter the cost for traveling from node i
to node j, i.e., the cost for traversing (i,j).
      c(1,3) = 7
      c(1,4) = 12
      c(2,1) = 12
      c(2,3) = 5
      c(2,7) = 12
      c(2,8) = 21
      c(3,2) = 5
      c(3,7) = 10
      c(4,1) = 12
      c(4,5) = 7
      c(4,9) = 2
      c(5,3) = 3
      c(5,1) = 6
      c(5,4) = 7
      c(5,9) = 13
      c(5,6) = 15
      c(6,5) = 8
      c(6,7) = 8
      c(6,8) = 7
      c(7,6) = 4
      c(7,8) = 9
      c(8,6) = 11
     c(8,12) = 1
     c(8,13) = 17
      c(9,5) = 9
     c(9,11) = 9
     c(9,12) = 16
     c(10,4) = 7
    c(10,11) = 4
    c(10,13) = 18
     c(11,9) = 14
    c(11,10) = 5
    c(11,12) = 3
    c(12,11) = 3
     c(12,8) = 5
```

Number of iterations = 11. Optimal set of nodes are as follows:

From node	Go to node	Min cost to terminal(13)
1	3	43.00
2	8	38.00
3	7	36.00
4	9	32.00
5	6	39.00
6	8	24.00
7	8	26.00
8	13	17.00
9	11	30.00
10	13	16.00
11	10	21.00
12	8	22.00

Output buildup of optimal solution? Answer(Yes=1;No=2) = 1

Iteration number = 1. State of the system follows:

From node	Go to node	Present Cost
1	3	7.00
2	3	5.00
3	2	5.00
4	9	2.00
5	3	3.00
6	8	7.00
7	6	4.00
8	12	1.00
9	11	9.00
10	11	4.00
11	12	3.00
12	11	3.00

Iteration number = 2. State of the system follows:

From node	Go to node	Present Cost
1	3	12.00
2	3	10.00
3	2	10.00
4	5	10.00
5	3	8.00
6	3	8.00
7	8	10.00
8	12	4.00
9	11	12.00
10	11	7.00
11	12	6.00
12	8	6.00

Iteration number = 3. State of the system follows:

From node	Go to node	Present Cost
1	3	17.00
2	3	15.00
3	2	15.00
4	9	14.00
5	3	13.00
6	8	11.00
7	6	12.00
8	12	7.00
9	11	15.00
10	11	10.00
11	12	9.00
12	8	9.00

Iteration number = 4. State of the system follows:

From node	Go to node	Present Cost
1	3	22.00
2	3	20.00
3	2	20.00
4	9	17.00
5	3	18.00
6	8	14.00
7	6	15.00
8	12	10.00
9	11	18.00
10	11	13.00
11	12	12.00
12	8	12.00

Iteration number = 5. State of the system follows:

From node	Go to node	Present Cost
1	3	27.00
2	3	25.00
3	7	25.00
4	9	20.00
5	3	23.00
6	8	17.00
7	6	18.00
8	12	13.00
9	11	21.00
10	11	16.00
11	12	15.00
12	8	15.00

Iteration number = 6. State of the system follows:

From node	Go to node	Present Cost
1	4	32.00
2	7	30.00
3	7	28.00
4	9	23.00
5	4	27.00
6	8	20.00
7	6	21.00
8	12	16.00
9	11	24.00
10	13	16.00
11	12	18.00
12	8	18.00

Iteration number = 7. State of the system follows:

From node	Go to node	Present Cost
1	4	35.00
2	7	33.00
3	7	31.00
4	9	26.00
5	4	30.00
6	8	23.00
7	6	24.00
8	13	17.00
9	11	27.00
10	13	16.00
11	12	21.00
12	8	21.00

Iteration number = 8. State of the system follows:

From node	Go to node	Present Cost
1	4	38.00
2	7	36.00
3	7	34.00
4	9	29.00
5	4	33.00
6	8	24.00
7	8	26.00
8	13	17.00
9	11	30.00
10	13	16.00
11	10	21.00
12	8	22.00

Iteration number = 9. State of the system follows:

From node	Go to node	Present Cost
1	4	41.00
2	8	38.00
3	7	36.00
4	9	32.00
5	4	36.00
6	8	24.00
7	8	26.00
8	13	17.00
9	11	30.00
10	13	16.00
11	10	21.00
12	8	22.00

Iteration number = 10. State of the system follows:

From node	Go to node	Present Cost
1	3	43.00
2	8	38.00
3	7	36.00
4	9	32.00
5	6	39.00
6	8	24.00
7	8	26.00
8	13	17.00
9	11	30.00
10	13	16.00
11	10	21.00
12	8	22.00

Set T = 5.

Do part 2.

Number of iterations = 9. Optimal set of nodes are as follows:

From node	Go to node	Min cost to terminal(5)
1	4	19.00
2	7	24.00
3	7	22.00
4	5	7.00
6	5	8.00
7	6	12.00
8	6	19.00
9	5	9.00
10	4	14.00
11	10	19.00
12	11	22.00

Output buildup of optimal solution? Answer(Yes=1;No=2) = 2

Appendix B

PROGRAM LISTING

```
Use file 238 (SYOP8).
Roger.
Recall item 1 (DYNAM).
Done.
Type all,size.

1.10  Demand N as "Number of nodes in the system".
1.11  Demand L as "Lowest node number in the system".
1.12  Demand T as "Terminal node number".
1.15  Demand q as "Are node numbers consecutive? (Answer:Yes=1;No=2)".
1.17  Do part [q=1:10;11].
1.20  Type "Into array A(i), enter number of arcs leading out of node i.".
1.24  Do part 12 for i=1(1)N.
1.30  Type "Into array O(i,j), enter node number into which the j-th arc".
1.31  Type "out of node i goes.".
1.34  Do part 13 for i=1(1)N.
1.36  Demand q as "Are costs to be entered here? (Yes=1;No=2)".
1.37  To part 2 if q=2.
1.40  Type "Into array c(i,j), enter the cost for traveling from node i".
1.41  Type "to node j, i.e., the cost for traversing (i,j).".
1.44  Do part 14 for i=1(1)N.
1.70  To part 2.

2.09  Set e=1.
2.10  Set k=0.
2.11  Set f=0.
2.15  Set k=k+1.
2.16  Set h=k-1.
2.17  Do part 20 if k>2 and (size-1700)>0.
2.18  Do part 3 for i=1(1)N.
2.20  Do part 5 for i=1(1)N.
2.24  To step [f=0:2.15;6.10].

3.10  Set y(k,n(i))=[n(i)=T:0;Q].
3.20  Do part 4 for j=1(1)A(n(i)).

4.10  Done if [y(h,O(n(i),j))+c(n(i),O(n(i),j))]=y(k,n(i)).
4.13  Set g(k,n(i))=[O(n(i),j)=T:T;g(k,n(i))=T:g(k,n(i));O(n(i),j)].

5.10  Quit if y(k,n(i))=y(k-1,n(i)).
5.13  Done if i=N.
5.15  Set f=1.
```

```
6.10  Page.
6.11  Type k in form 1.
6.13  Line.
6.14  Type T in form 2.
6.15  Type form 3.
6.16  Do part 7 for i=1(1)N.
6.17  Type _,_,_.
6.20  Demand q as "Output buildup of optimal solution? Answer(Yes=1;No=2)".
6.25  Do part 8 for z=1(1)h if q=1.

7.10  Done if n(i)=T or A(n(i))=0.
7.15  Type n(i),g(k,n(i)),y(k,n(i)) in form 4.

8.01  Type _,_,_.
8.02  Type z in form 5.
8.03  Type form 6, form 3.
8.04  Do part 9 for i=1(1)N.

9.01  Done if n(i)=T or A(n(i))=0.
9.02  Type n(i),g(z,n(i)),y(z,n(i)) in form 4.

10.1  Do step 10.5 for i=1(1)N.
10.2  Done.
10.5  Set n(i)=i+L-1.

11.1  Type "Into array n(i). enter number of the ith node.".
11.2  Do step 11.5 for i=1(1)N.
11.3  Done.
11.5  Demand n(i).

12.1  Demand A(n(i)).

13.1  Done if A(n(i))≤0.
13.2  Do step 13.5 for j=1(1)A(i).
13.3  Done.
13.5  Demand O(n(i),j).

14.1  Done if A(i)≤0.
14.2  Do step 14.5 for j=1(1)A(i).
14.3  Done.
14.5  Demand c(n(i),O(n(i),j)).

20.1  Set b=k-2.
20.2  Do part 21 for d=b(-1)e.
20.3  Set e=b+1.
20.4  Delete part 1,part 8,part 9,step 6.20,step 6.25,step 20.4.

21.1  Do part 22 for i=1(1)N.

22.1  Done if n(i)=T.
22.2  Delete g(d,n(i)), y(d,n(i)).

25.1  To step 1.44.
```

Form 1:
Number of iterations =____. Optimal set of nodes are as follows:

Form 2:
From node Go to node Min cost to terminal(___)

Form 3:
--------- ---------- ------------------------

Form 4:

_____ _____ _____.__

Form 5:
Iteration number =_____. State of the system follows:

Form 6:
From node Go to node Present Cost

Q: [A(n(i))≤C:10^6; min[j=1(1)A(n(i)):y(h,O(n(i),j))+c(n(i),O(n(i),j))]]

g(0,0) = 0
g is sparse

y(0,0) = 0
y is sparse

size = 513

REFERENCES

1. Dreyfuss, S. E., *An Appraisal of Some Shortest-Path Algorithms*, The Rand Corporation, RM-5431-PR, September 1968.

2. Ford, L. R., and D. R. Fulkerson, *Flows in Networks*, The Rand Corporation, R-375-PR, August 1962.

3. Wagner, Harvey M., *Principles of Operations Research with Applications to Managerial Decisions*, Prentice-Hall, Inc., New Jersey, 1969.

4. Bryan, G. E., and E. W. Paxson, *The JOSS Notebook*, The Rand Corporation, RM-5367-PR, August 1967.

DOCUMENT CONTROL DATA

1. ORIGINATING ACTIVITY	2a. REPORT SECURITY CLASSIFICATION
The Rand Corporation	UNCLASSIFIED
	2b. GROUP

3. REPORT TITLE

A DYNAMIC PROGRAMMING APPROACH TO NETWORK PROBLEMS: A MODEL FOR ON-LINE COMPUTER SYSTEMS

4. AUTHOR(S) (Last name, first name, initial)

Pipes, L. J.

5. REPORT DATE	6a. TOTAL NO. OF PAGES	6b. NO. OF REFS.
August 1970	28	4
7. CONTRACT OR GRANT NO.	8. ORIGINATOR'S REPORT NO.	
DAHC15-67-C-0142	RM-6338-ARPA	
9a. AVAILABILITY/LIMITATION NOTICES	9b. SPONSORING AGENCY	
DDC 1	Advanced Research Projects Agency	

10. ABSTRACT

Derivation, documentation, and listing of a dynamic programming model on JOSS for finding optimal solutions (minimum-cost paths) to network problems. The network represents the possible connections between decisions and expresses costs of going from one to another. The model encompasses a value-iterative algorithm which successively converges to the solution. The principle of optimality, which permits finding the minimum-cost (shortest) path as a function of the maximum number of arcs allowed, is: Regardless of how a particular state was arrived at, the remaining decisions as to what path to take enroute to the terminal must themselves constitute an optimal solution. The costs of traversing arcs may be input, recalled from other files, or calculated by JOSS from input functions. Model flexibility allows for testing the sensitivity of a decision process to changes in the terminal point.

11. KEY WORDS

JOSS (Interactive Computing System)
Networks
Decisionmaking
Mathematical Programming
Combinatorics